UNIVERSITY OF QUEENSLAND
COMPUTER CENTRE

COMPUTER CENTRE

BULLETIN

Volume 2, Number 2.
3rd February, 1969.

Editor:
H. L. Smythe.

THIS EDITION

This issue of the Bulletin introduces Professor G.A. Rose who was recently appointed to the position of Professor of Computer Science.

There is an article describing a Digital Analogue-Simulation Language, and a new section lists some of the library accessions which may be of interest to the reader.

Also included is an article describing card interpretation, a new facility available at the Computer Centre.

Other articles discuss new library programs and systems modification, and give programming advice.

PROFESSOR OF COMPUTER SCIENCE

Professor Rose has been appointed to the newly-established Chair in Computer Science and takes up duty on 1st March, 1969. He graduated from the University of Adelaide in 1952 with First Class Honours, and subsequently had extensive professional experience as a Scientific Officer with the Department of Supply (in Australia and the United Kingdom).

In 1958, he was appointed Lecturer and later Senior Lecturer in Electrical Engineering at the University of Adelaide. During the period 1959 to 1962, he was responsible for the detailed specifications and design of the arithmetic and control sections of the CIRRUS digital computer project. CIRRUS was one of the first multiprogrammed machines ever constructed, and several advanced microprogramming techniques were developed. This interest was expanded during a year's Study Leave in U.S.A.

In September 1965, he joined the staff of the University of New South Wales where he initiated and was a principal contributor to the intergraphic project. This system has been constructed from high speed integrated circuits controlled by microprogram, and has made a major contribution to computer engineering as the first low cost graphical communication system for arbitrary graphics. He presented a paper on this subject to the International Federation of Information Processing Societies in

Edinburgh, August 1968.

Professor Rose brings to the University of Queensland extensive experience in computer languages, and will attach much importance to the development of the software associated with the present system. His object as Head of the Department of Computer Science is to coordinate the teaching and research activities with the development of the present service functions of the Computer Centre. He will also seek to maintain a balance between the "Centre directed" research of the Department and research in other aspects. He hopes to develop dynamic graphical facilities within the Department, but this is likely to await completion of the present development of batch processing and remote terminal operation of the PDP 10 system.

The University welcomes Professor Rose, and looks forward to a period of even more rapid development of teaching and research in the field of computing. Computer Science has full status as a University Department from 1st January, 1969; and, until Professor Rose can take up duty, Professor Prentice is Acting Head.

CARD INTERPRETATION

Equipment is now available at the Computer Centre for the "interpretation" of alphanumeric cards. The process of interpretation consists of sensing the holes punched in the cards and printing the corresponding graphics along the top of these cards.

A common practice is to re-sequence-number a deck with one of the Centre's "Sequence Numbering Reproducer" programs, B6.200 or B6.201, and then interpret the resequenced deck that is produced.

Decks to be interpreted should be presented, together with the usual Data Preparation Requisition Form, to the Administrative Officer. Charges for interpretation are the same as those for keypunching, since an 029 card punch which has this additional feature, is the device used for interpreting cards. About 780 fully-punched cards are interpreted per hour.

Note that the graphics printed on the cards correspond to the 029 character set rather than to the 026. Thus, in the case of cards punched with the 026 character set, the graphics produced for the characters + = ( ) ' will be & # % < @ respectively.

PROGRAMMING ADVICE

MAGNETIC TAPES

When FORTRAN IV is used to write BCD information on magnetic tape, it tests for, and *deletes* any null records. Hence the following statements:

```
           WRITE (7, 10)
    10     FORMAT (9HTAPE NAME / / 9HFILE NAME)
```

act as though the FORMAT statement were:

```
    10     FORMAT (9HTAPE NAME / 9HFILE NAME)
```

The format specification, 1HΔ, can be used to obtain a blank record. For the above example, the FORMAT statement would be:

```
    10     FORMAT (9HTAPE NAME / 1HΔ / 9HFILE NAME)
```

SYSTEM MODIFICATION

AINT FUNCTION - FORTRAN IV

The FORTRAN IV Library function AINT was found to be in error and has been corrected. Previously, the function "fixed" the argument to obtain the integer part, but did not "float" it, so that it returned a fixed point result rather than a floating point result. The calling program then attempted to interpret the result as a floating point number.

LIBRARY PROGRAMS

NEW

MAIL - Mailing List Package (K1.206)

A mailing list package has been developed to create and maintain a magnetic tape file of names and addresses, so that address labels may be printed for the circulation of routine correspondence, circulars, etc. Each entry is composed of an identification code, name and address details, and a mailing code which allows the one file to be used for as many as 18 classes of circulation. As the mailing code comprises a single digit for each of the 18 classes, it may also be used to signify the number of items of the

class to be forwarded to a particular addressee.

The label writing programs will, by means of a parameter card, print labels for all entries in the selected classes.

The package is written in GECOM and comprises the following programs -

MAILTAPE - (K1.206A) Creates the basic tape file, with proper labels.

MAILUPDT - (K1.206B) Adds entries to an existing mail file.

MAILLABLI - (K1.206C) Prints address labels without class codes.

MAILLABLII - (K1.206D) Prints address labels with class codes.

MAILLIST - (K1.206E) Lists the mailing file with entry numbers and summary.

MAILEDIT - (K1.206F) Edits the mailing file.

As a matter of interest, the mailing labels for the Bulletin are now produced by this package.

LIBRARY ACCESSIONS

This new section of the Bulletin will keep readers informed of the books and periodicals, concerning the computer field, that have recently been acquired by Libraries of the University of Queensland. They are selected from the University of Queensland Library Accession List only by title, and there is no guarantee that the following list is complete or that the material is of high quality.

The following list comprises the accessions for the months of July and August, 1968:

Bergstrom, Abram R. *The Construction and use of Economic Models.* [c1967] (330.0182 BER, That. Mem. Lib.)

Gt. Brit. Ministry of Technology. *Electronic Automation in Britain: 1966.* 1966. (Q338.064 GRE, Arch. Lib.)

Integrated Process Control Applications in Industry. [1966?] (Q338.064 INT, Elec. Eng. Lib.)

Szabo, Nicholas S. *Residue Arithmetic and its Applications to Computer Technology.* [c1967] (512.813 SZA, Engin. Lib.)

Queensland University. *Computer Centre Bulletin.* v. 1 1968 and onwards. (510.7834 QUE, Uni. P.)

Martin, James Thomas. *Design of Real-Time Computer Systems.* 1967.
(651.8 MAR, Engin. Lib.)

Pinkney, A. *An Audit Approach to Computers.* 1966. (657.6 PIN, Main Lib.)

Shuchman, Abraham Comp. *Scientific Decision-making in Business.* 1963.
(658 SHU, Main. Lib.)

Convention on Advances in Computer Control. University of Bristol. 1967.
Advances in Computer Control, 1967. (Q626.8 CON, Elec. Eng. Lib.)

International Business Machines Corporation. *Introduction to IBM Data Processing Systems.* [c1967] (Q651.8 INT, Account S.R. Lib.)

Myers, Charles Andrew. ed. *The Impact of Computers on Management.* [c1967]
(658 MYE, Main Lib.)

DIGITAL-ANALOGUE SIMULATION - L. Mor

INTRODUCTION

Many engineering systems can be described in terms of time-varying linear differential equations, and solutions to these are often more easily obtained using an analogue computer than by using other methods, such as numerical integration. As an analogue computer effectively 'simulates' the system under study, considerable importance can be attached to sub-systems within the simulation, and, in this way, the engineer obtains a better understanding of the interactions of the system. In addition, having established the mathematical model on the analogue computer, it is relatively simple to modify parameters of the system and to view the effects. This type of experimentation is often expensive and even impossible to carry out with the real-life system.

With the current popularity of digital computers and the ready availability of large, fast machines, it is desirable to make the analogue techniques available to digital machine users. Programs have therefore been written to provide this simulation procedure on a digital computer in order to combine some of the advantages of analogue computation with the flexibility of the digital machine. Such programs have the following advantages:

(a) It is possible, using only a medium-sized digital computer, to apply this technique to a problem which otherwise would need a large analogue computer.

(b) It is relatively simple to simulate non-linear operations and time delays. In addition, other facilities of the digital computer are naturally available to the user.

(c) There are no amplitude scaling problems.

(d) Greater accuracy can be obtained.

(e) Problems of noise and drift do not occur.

Yet there remains the disadvantage that the generation of solutions for large systems takes longer on the digital computer. However, the evidence of recent articles shows digital simulation to be a new and powerful tool for engineers and mathematicians.

This article describes a digital simulation language which is being developed by the author using the GE 225 Computer. Since this machine operates in a batch processing environment, there can be no user interaction. However, the language has also been designed for use in a time-sharing system where interaction is possible.

THE SIMULATION LANGUAGE

Introduction

The simulation language is a problem-oriented language which can be used for investigating systems mathematically described by a set of differential equations. It is expressly designed for engineers and mathematicians with no digital programming experience, and, although oriented towards the analogue programmer, it is a basic and easily-learnt language. Programs are written in meaningful grammatical statements in analogue terminology.

Compilation of the simulation source program by the compiler generates a FORTRAN IV program, and, provided no errors are detected during compilation, the object program is automatically run as a compile-and-go FORTRAN program.

On the GE 225, it will be possible only to output results on the high speed printer in a tabular or graphical form, but, in a time-sharing environment, additional options will be made available.

The compiler contains a comprehensive set of diagnostics to aid the user in debugging programs.

Description

Where possible, the language attempts to simulate the processes which would be used in developing a model on the analogue computer. Consequently, each program consists of sections which are written in the following order:

Identification Section	- identifies the user and the problem
Definition Section	- defines the number and type of computing elements
Patching Section	- patches the elements defined
Description Section	- describes the method of output of results
Specification Section	- specifies initial values for all elements and various program parameters
Repetition Section	- provides a repetition facility. This section enables a user to re-patch elements, re-describe output or reset initial values.

The first five sections are mandatory. Each section contains a number of statements which can be inserted in any order within the section, and are written in "free-field" format.

Within the context of a program, certain character strings or words have special significance to the compiler.

(a) Key Words

These are words which have special significance to the compiler, and, because of this, may not be referenced by the user as variable names. Key words are section, control and noise words, and element names.

(b) Section Words

Section words are key words which are placed at the beginning of each new section of the program to delimit sections effectively.

(c) Control Words

These are words which instruct the compiler to process the information following the instructions of the control words. Each statement contains one control word (in fact, this is used to delimit statements), and each control word is only legal within its section of definition. The special control words 'REMARK' or 'REMARKS' enable the user to insert comments anywhere within a program.

(d) Noise Words

Noise words are words which may be inserted in a program at will to improve the overall intelligibility.

(e) Elements

An element is the basic structural unit of the simulation configuration. Each element e.g. integrator, adder, etc. may have zero, one, or more inputs, but has only one output, although this may be connected to a number of other elements. The convention used is that the output of each element in the simulation must be defined by a unique variable name.

In the basic form of the language, the following elements are available:

ADDER(S)	INVERTER(S)
CONSTANT(S)	LIMITER(S)
DELAY(S)	MULTIPLIER(S)
DIVIDER(S)	POTENTIOMETER(S)
EXPONENTIATOR(S)	RELAY(S)
FUNCTION(S)	SUBTRACTOR(S)
INTEGRATOR(S)	

Both singular and plural forms of the element names are admissible, and all names are reserved words. Other elements will be added as the need arises.

The element FUNCTION, enables the user to define any special purpose element. The properties of the function element are defined in the form of a standard FORTRAN IV real function sub-program.

Unlike the analogue computer, there is no sign change associated with the 'active' elements, and potentiometers can have a coefficient greater than 1.0 with positive or negative sign.

(f) Variable Names

Variable names are words used to define the elements of the simulation. The variable names so defined are used in the generated FORTRAN program, and hence must conform to FORTRAN specifications.

Method of Integration

The integrator element uses a third order predictor method of

integration, and the accuracy obtained with a reasonable value of integration interval is sufficient for most engineering applications. However, if greater accuracy is required, a standard 4th order Runge-Kutta integration subroutine is also available.

Example

The program in Example 4 illustrates some of the features of the language. It solves a second order linear differential equation with no damping, generating the sine (SINT) and cosine (COST) of the independent variable time (TIME). These variables are squared and summed to produce SUMSQUARES. In addition, the program generates for comparison the actual sine (ACTSINT) and cosine (ACTCOST) of time using the FORTRAN IV library subroutines SIN and COS. The variables COST, SINT, SUMSQUARES, ACTCOST and ACTSINT are printed as a function of TIME.

Example 5 shows the second set of results produced by the program.

```
IDENTIFICATION SECTION.
     CODE 935/01, RUNTIME 10 MINUTES.

REMARKS  PROBLEM 3  PROGRAM TO TEST ACCURACY OF THE INTEGRATION SUBROUTINES

DEFINITION SECTION.
     DEFINE INTEGRATORS COST, SINT.
     FUNCTION SIN ACTSINT.
     FUNCTION COS ACTCOST.
     MULTIPLIERS SQCT, SQST.
     ADDER SUMSQUARES.
     INVERTER MINUS1.

PATCHING SECTION.
     PATCH COST TO SINT TO MINUS1 TO COST.
     PATCH COST TO SQCT.
     PATCH COST TO SQCT.
     PATCH SINT TO SQST.
     PATCH SINT TO SQST.
     PATCH SQCT TO SUMSQUARES.
     PATCH SQST TO SUMSQUARES.
     PATCH TIME TO ACTSINT.
     PATCH TIME TO ACTCOST.

DESCRIPTION SECTION.
      PRINT TIME, COST, SINT, SUMSQUARES, ACTCOST, ACTSINT.
     HEAD  SOLUTION OF A SECOND ORDER LINEAR DIFFERENTIAL EQUATION
     HEAD  WITH NO DAMPING
     HEAD  INTEGRATION INTERVAL 1/10TH NATURAL PERIOD

SPECIFICATION SECTION.
     SET COST= 1.0, SQCT= 1.0, SUMSQUARES= 1.0, ACTCOST= 1.0.
     RUN FOR 10.0, INCREMENT BY 0.628, OUTPUT EVERY 0.628.

REPETITION SECTION.
     REPEAT.
     RESET INCREMENT TO 0.0628.
     REHEAD   INTEGRATION INTERVAL 1/100TH NATURAL PERIOD

END
```

Example 4

INTEGRATION INTERVAL 1/100TH NATURAL PERIOD

TIME	COST	SINT	SUMSQUARES	ACTCOST	ACTSINT
0.00000E-01	1.00000E+00	0.00000E-01	1.00000E+00	1.00000E+00	0.00000E-01
6.28000E-01	8.09204E-01	5.87527E-01	1.00000E+00	8.09204E-01	5.87528E-01
1.25600E+00	3.09623E-01	9.50859E-01	1.00000E+00	3.09623E-01	9.50859E-01
1.88400E+00	-3.08108E-01	9.51351E-01	1.00000E+00	-3.08108E-01	9.51351E-01
2.51200E+00	-8.08267E-01	5.88816E-01	1.00000E+00	-8.08267E-01	5.88816E-01
3.14000E+00	-9.99999E-01	1.59305E-03	1.00000E+00	-9.99999E-01	1.59289E-03
3.76800E+00	-8.10139E-01	-5.86238E-01	1.00000E+00	-8.10139E-01	-5.86238E-01
4.39600E+00	-3.11137E-01	-9.50365E-01	1.00000E+00	-3.11137E-01	-9.50365E-01
5.02400E+00	3.06592E-01	-9.51841E-01	1.00000E+00	3.06592E-01	-9.51841E-01
5.65200E+00	8.07328E-01	-5.90103E-01	1.00000E+00	8.07328E-01	-5.90108E-01
6.28000E+00	9.99995E-01	-3.18612E-03	1.00000E+00	9.99995E-01	-3.18608E-03
6.90800E+00	8.11072E-01	5.84946E-01	1.00000E+00	8.11072E-01	5.84946E-01
7.53600E+00	3.12651E-01	9.49868E-01	1.00000E+00	3.12651E-01	9.49868E-01
8.16400E+00	-3.05075E-01	9.52328E-01	1.00000E+00	-3.05075E-01	9.52328E-01
8.79200E+00	-8.06387E-01	5.91388E-01	1.00000E+00	-8.06387E-01	5.91388E-01
9.42000E+00	-9.99989E-01	4.77917E-03	1.00000E+00	-9.99989E-01	4.77916E-03
1.00480E+01	-8.12003E-01	-5.83653E-01	1.00000E+00	-8.12003E-01	-5.83658E-01

Example 5